Finding Structurally and Temporally Similar Trajectories in Graphs

Roberto Grossi, Andrea Marino, Shima Moghtasedi

Finding Structurally and Temporally Similar Trajectories in Graphs

Roberto Grossi
Dipartimento di Informatica, Università di Pisa, Italy
grossi@di.unipi.it

Andrea Marino
Dipartimento di Statistica, Informatica, Applicazioni "G. Parenti", Università di Firenze, Italy
andrea.marino@unifi.it

Shima Moghtasedi
Dipartimento di Informatica, Università di Pisa, Italy
shima.moghtasedi@di.unipi.it

Abstract

The analysis of similar motions in a network provides useful information for different applications like route recommendation. We are interested in algorithms to efficiently retrieve trajectories that are similar to a given query trajectory. For this task many studies have focused on extracting the geometrical information of trajectories. In this paper we investigate the properties of trajectories moving along the paths of a network. We provide a similarity function by making use of both the temporal aspect of trajectories and the structure of the underlying network. We propose an approximation technique that offers the top-k similar trajectories with respect to a query trajectory in an efficient way with acceptable precision. We investigate our method over real-world networks, and our experimental results show the effectiveness of the proposed method.

2012 ACM Subject Classification Information systems → Similarity measures; Information systems → Nearest-neighbor search

Keywords and phrases Graph trajectory, approximated similarity, top-k similarity query

Digital Object Identifier 10.4230/LIPIcs.SEA.2020.24

Acknowledgements We are in debt with Ioanna Miliou for helping us with the Milan GPS dataset.

1 Introduction

Many papers in the literature have focused on extracting similarity information from sets of trajectories [1, 4, 3, 5, 7, 9, 16, 11, 12, 13, 14, 8, 15, 16]. Looking at the trajectories as sequences of nodes, the similarity among them can be related to the similarity among sequences. In this paper, we study how to retrieve similar trajectories constrained to follow paths in the graphs by taking into account both time and place. We aim at exploiting the topology of the network, assessing that two trajectories are similar if they pass through nearby nodes at roughly the same time. While there are measures such as the Fréchet distance for the plane, not much has been done for graphs. Indeed distances on the plane are fast to compute as we need to know just the coordinates of the points, so that measures taking into account both time and place can be computed in a reasonable time.

On the other hand, when trajectories are topologically constrained, as it happens in graphs, the distance computation between nodes is more challenging and the similarity function can turn easily into measures that are costly to compute. For this reason, the measures in the known literature that consider both time and place require that similar trajectories should pass through the same nodes. Table 1 summarizes the state of the art for the similarities of trajectories in graphs. As it can be noted, most of them require

Table 1 Graph-based similarity measures for trajectories of ℓ nodes.

	Temporal	Proximity	Properties	Input	Complexity time
[15]	×	×	Jaccard similarity based	Two strings as trajectories	$O(\ell^2)$
[16]	✓	×	Jaccard similarity based on the edit distance	Two strings as trajectories	$O(\ell^2)$
[8]	✓	✓	Pair to pair distance computation only at **specific predefined points**	Two trajectories with **the same length**, implicitly	$\boldsymbol{O(\ell)}$
[12]	✓	✓	Spatial and temporal distance computation in separate way (Liner combination)	Two trajectories with **the same length**	$\boldsymbol{O(\ell)}$
[11]	✓	✓	LCSS based Liner combination of spatial and temporal distance	Two trajectories	$O(\ell^2)$
[13]	✓	✓	Linear combination of spatial and temporal distance	A set of query points and a trajectory	$O(\ell^2)$
This paper	✓	✓	Temporal and spatial aspect of trajectories is combined in one single distance	Two trajectories with **different length**	$\boldsymbol{O(\ell)}$

quadratic time and those requiring linear time [1] have limitations: Hwang et al. [8] define the spatial-temporal similarity function so that two trajectories are similar if they pass through the same nodes at the same time. So they do not take into account the proximity of the trajectories and the closure in time of the intervals. Tikas et al. [12] remove this limitation, proposing a new similarity definition. However, their similarity works only when trajectories have the same length. As far as we know, there is no linear-time similarity that considers trajectories of different lengths and with a flexible notion of proximity in time and place.

In this paper, we show that it is possible to consider both time/structure for the trajectories on the graph, thus overcoming the limitations of the current literature and achieving a good compromise between time and precision. After defining our similarity function, we propose an algorithm to find the topmost k trajectories, having the highest similarity with respect to a given one. We do not put any constraint on the trajectories, and our similarity function can be computed in linear time once the pairwise node distances are given.

Our method is based on an indexing structure by using interval trees [2] to quickly find the top-k similar trajectories. To further speed up the computations, we propose an approximated similarity measure that shrinks the trajectories using the centers of the Voronoi diagram for graphs [6]. Shrinking using centers is natural as, for example, a long trajectory that goes along a road network can be reasonably represented with the most famous visited places. Similarly, the trajectories on the communication networks could be represented with respect to the nodes having more traffic.

We validate our measures and algorithms in the experimental part of the paper. Due to the lack of competitors, as there are no linear-time similarities on trajectories for graphs that use flexible proximity, we design a baseline approach to compare with. The approximation methods have a good precision while the time needed to answer the query reduces significantly.

[1] We are assuming for all of them that the pairwise distance among nodes is already given.

2 Preliminaries

Consider a network represented by a graph $G(V, E)$, where V is the set of nodes and E is the set of edges. Let $\mathcal{T}$ be a set of trajectories in G, where each trajectory is defined as follows. We say that two intervals $t_i = [s_i, e_i]$ and $t_{i+1} = [s_{i+1}, e_{i+1}]$, with integer endpoints, are *consecutive* if $s_i \leq e_i < s_{i+1} \leq e_{i+1}$ and $e_i + 1 = s_{i+1}$.

▶ **Definition 1.** *A trajectory $T \in \mathcal{T}$ is a sequence $T = \langle (v_1, t_1), (v_2, t_2), \cdots, (v_l, t_l) \rangle$ such that for each $1 \leq i \leq l-1$, we have that $(v_i, v_{i+1}) \in E$ and t_i and t_{i+1} are two consecutive time intervals. We call $|T| = l$ the length of T, which corresponds to the number of (non-distinct) nodes traversed by T. Letting $t_1 = [s_1, e_1]$ and $t_l = [s_l, e_l]$, we refer to s_1 and e_l as the starting time and ending time of T.*

Given a trajectory $T \in \mathcal{T}$, we denote by $t_i = [s_i, e_i]$ the i-th time interval of T. Note that a trajectory can pass through a node more than once.

For a trajectory T, let s and e be the starting and ending time of T. Given a time instant $i \in [s, e]$, the notation $T(i)$ indicates the unique node $u \in V$ such that there exists a pair $(u, t) \in T$ with $i \in t$. In the following, we use the standard notation for graphs. Given an undirected graph $G = (V, E)$, we denote by n and m the number of its nodes and edges. We denote by D_G the diameter of G and by $d(u, v)$ the shortest path distance between nodes u and v.

2.1 Trajectory Similarity Measure

This section is devoted to introducing our similarity function. It is arguably natural to assess that two trajectories are *similar* if they are close to each other without necessarily sharing common nodes or having the same length. Since the motion of trajectories in this paper is constrained by the network, the similarity function will use the proximity between trajectories. As the Euclidean distance is not appropriate to measure the distance between the nodes on the graph, it is important to use the graph distance metric instead. Therefore, the distance between nodes will be combined with the time intervals at which these have been traversed. In this sense, our similarity measure will consider both aspects of the trajectories: the *temporal* aspect and the location of trajectories over the graph, i.e the *structural* aspect.

In order to define the building blocks of our similarity measure, we need first to restrict trajectories within a time interval, as shown in the following definition.

▶ **Definition 2** (Time restricted Trajectory)**.** *Given a trajectory T and a time interval $t = [s, e]$, the time restricted trajectory $T[t]$ is the sequence of pairs $(v_i, t_i) \in T$ such that $t_i = [s_i, e_i]$ has overlap with $t = [s, e]$ (i.e. $t_i \cap t \neq 0$).*

Without loss of generality, we assume that $\sum_{(v_i, t_i) \in T[t]} |t_i| = |t|$. We define the distance between a *node* v and a *trajectory* T within a *time interval* t as follows:

$$dist(v, T, t) = \frac{\min_{(v_i, t_i) \in T[t]} d(v_i, v)}{D_G} \tag{1}$$

▶ **Proposition 3.** *The distance function $dist(v, T, t)$ is always in the interval $[0, 1]$.*

We observe that the extreme values are achieved in the following cases.

- $dist(v, T, t) = 0$ if and only if there exists at least one time instant $i \in t$ such that node $T(i) = v$.

- $dist(v, T, t) = 1$ if and only if for each time instant $i \in t$, node $T(i)$ is at distance D_G from v. This corresponds to the case where T spends the whole time interval t on nodes of G that are maximum distance from v.

We are now ready to introduce our similarity measure, using the distance function defined in Equation 1. Taking inspiration from [5], we aim at assigning a larger contribution to those parts of trajectories that are close for sufficiently long time intervals (while assigning lower contribution to farther parts). These desired properties are satisfied as follows.

▶ **Definition 4** (Similarity Function). *Given a query trajectory Q, a target trajectory T, and a time interval t, the similarity of T with respect to Q within t is*

$$Sim(Q, T, t) = \frac{\sum_{(v_i,t_i)\in Q[t]} |t_i| \times e^{-dist(v_i,T[t_i])}}{|t|} \tag{2}$$

▶ **Lemma 5.** *The similarity function $Sim(Q, T, t)$ is always in the interval $(0, 1]$.*

Proof. By Proposition 3, for each $(v_i, t_i) \in Q[t]$ we have that $0 \leq dist(v_i, T[t_i]) \leq 1$, and thus $1 \geq e^{-dist(v_i,T[t_i])} \geq e^{-1} > 0$. If we multiply by $|t_i|$, we get $0 < |t_i| \times e^{-dist(v_i,T[t_i])} \leq |t_i|$. By summation over each pair $(v_i, t_i) \in Q[t]$ we get

$$0 < \sum_{(v_i,t_i)\in Q[t]} |t_i| \times e^{-dist(v_i,T[t_i])} \leq \sum_{(v_i,t_i)\in Q[t]} |t_i| \tag{3}$$

By assuming $\sum_{(v_i,t_i)\in Q[t]} |t_i| = |t|$ and dividing Equation 3 by $|t|$ we get

$$0 < \frac{\sum_{(v_i,t_i)\in Q[t]} |t_i| \times e^{-dist(v_i,T[t_i])}}{|t|} \leq 1. \qquad \blacktriangleleft$$

It is worth remarking that whenever $Q[t] = T[t]$ we have $Sim(Q, T, t) = 1$, where $Q[t] = T[t]$ means that for each $i \in t$ we have $Q(i) = T(i)$.

▶ **Lemma 6.** *Given two trajectories Q and T, and a time interval t, where $|Q[t]| = \ell_1$ and $|T[t]| = \ell_2$, computing $Sim(Q, T, t)$ requires $O(\ell_1 + \ell_2)$ time and pairwise node distances.*

Proof. Looking at equations (1) and (2), it seems that $O(\ell_1 \times \ell_2)$ time is needed. The cost is instead $O(\ell_1 + \ell_2)$ if we realize that the computation is conceptually a nested loop in which the nodes in Q and T are scanned forward when a pairwise distance $d(v_i, v)$ is needed: in each iteration at least one node is scanned, thus the total cost is $O(\ell_1 + \ell_2)$. ◀

2.2 Top-k Most Similar Trajectories

We define the problem of retrieving, in a given set of trajectories $\mathcal{T}$, the top-k similar trajectories to a query in a specific time interval. More formally, this desires a set of trajectories, referred to as k-MsTraj, corresponds to the following one.

▶ **Definition 7** (k-Most Similar Trajectory (k-MsTraj)). *Given a set of trajectories $\mathcal{T}$, a query trajectory Q, and a query time interval t, let k-MsTraj be the set $\mathcal{T}' \subseteq \mathcal{T}$ with $|\mathcal{T}'| = k$, such that $Sim(Q, S, t) \geq Sim(Q, T, t)$ for each trajectory $S \in \mathcal{T}'$ and $T \in \mathcal{T} - \mathcal{T}'$.*

To present a good intuition of our proposed similarity function, we provide an illustrative example of four trajectories that are randomly chosen from a dataset of trajectories moving in Milan (see Section 5.1 for more details about this dataset). By the similarity function in Definition 4, using the red trajectory in Figure 1 as query trajectory, the green trajectory

has the maximum similarity among all the others, meaning that it is a solution for the k-MsTraj problem with $k = 1$. Note that the trajectories having color red, green, yellow and violet in Figure 1, start to move at time instances (in msec) 37237, 45964, 57354 and 26430, and stop the movement at 582313, 331565, 57872 and 564740, respectively.

Figure 1 An example including 4 random trajectories in a dataset of trajectories moving in Milan. The trajectory with the red color is a query. The green trajectory is the most similar one by Definition 4.

A straightforward approach to find k-MsTraj is to compute the similarity score for each trajectory $T \in \mathcal{T}$ and Q, reporting the k trajectories with maximum scores. Clearly we only consider those trajectories that are defined for all instants $i \in t$. This approach is inefficient as it requires $O(|\mathcal{T}| \times \max\{|y|_{max}, |Q|\})$ shortest path (precomputed) distances, where $|\mathcal{T}|$ and $|y|_{max}$ are respectively the number of trajectories and the maximum length of trajectories in $\mathcal{T}$.

In the next two sections, we discuss how to accelerate this method and how to estimate similarities.

3 Baseline: Exact Computation of k-MsTraj

In this section, we introduce a baseline method to solve exactly the k-MsTraj problem. This is based on an indexing phase, described in Section 3.1, which aims at accelerating the query processing, as described in Section 3.2. We will use this method as a baseline in the experimental evaluation of our proposed methods.

3.1 NTrajI Indexing

The *Neighborhood Trajectory Indexing* (NTrajI) described here efficiently finds the closest trajectories with respect to each node of the query and its corresponding time interval.

We use an interval tree, which is a binary tree storing a set of intervals based on the median of the endpoints of the intervals. In this structure, all the intervals that intersect the median point are stored in the root of the tree. The intervals lying completely to the left and the right of the median point are, respectively, stored in the left subtree and the right subtree of the root. The subtrees are constructed recursively in the same way. By using this structure, we are able to find efficiently all intervals that overlap with any given interval or point using the following well-known result.

▶ **Theorem 8** ([2]). *Given a set of n intervals, an interval tree uses $O(n)$ space, can be built in $O(n \cdot \log n)$ time and can report all intervals that overlap a query interval or point in $O(\log n + k)$ time, where k is the number of reported intervals.*

In NTrajI, we build an Interval Tree IT_u for each node $u \in V$. Each IT_u stores the time intervals of the trajectories in $\mathcal{T}$ spent in either u or the neighbors of u, and maintains the corresponding trajectories. Specifically, we have the following.

▶ **Definition 9** (Node Projection Set). *The projection set S_u of a node u stores the pairs (t, T) of all trajectories $T \in \mathcal{T}$ that pass through the nodes in $\{u\} \cup N(u)$ during interval t, namely, $S_u = \{(t, T) \mid (v, t) \in T$ and $v \in \{u\} \cup N(u)$ and $T \in \mathcal{T}\}$, where $N(u)$ is the set of neighbors of u in the graph.*

The Interval Tree IT_u maintains all the pairs $(t, T) \in S_u$ for each node $u \in V$. Each entry of IT_u is of the form $\langle t, id \rangle$, where id is the trajectory identifier and t is the time interval spent in $\{u\} \cup N(u)$ by the trajectory id. Note that there can be more than one pair associated with node u and the same trajectory id, since each trajectory can traverse u multiple times.

By Theorem 8, we can derive that NTrajI uses $O(|\mathcal{T}| \times |y|_{max} \times \Delta)$ space, where Δ denotes the maximum degree of G. For a given node u and a time interval t, let $\Gamma_{(u,t)}$ denote the trajectories that traverse either u or $N(u)$ within t.By searching over the NTrajI, we are able to find $\Gamma_{(u,t)}$ efficiently by taking $O(\log |\Gamma_u| + |\Gamma_{(u,t)}|)$ time, where $|\Gamma_u|$ is the size of IT_u and $|\Gamma_{(u,t)}|$ is the number of reported trajectories.

Algorithm 1 Baseline.

Input: Graph G, set of trajectories $\mathcal{T}$, query trajectory Q, time interval $t = [a, b]$, integer k
Result: Top-k trajectories k-MsTraj

```
Heap H
for each (v_i, t_i) ∈ Q[t] do
    Γ_(v_i,t_i) ← NTrajI-search(v_i, t_i)
end for
Γ = ⋃_{(v_i,t_i)∈Q_t} Γ_(v_i,t_i)
H ← the Sim scores for all trajectories in Γ
k-MsTraj ← top-k trajectories in H
```

3.2 Query Processing

We introduce a pruning technique as the baseline method to the k-MsTraj problem (see Algorithm 1). The baseline method explores the set Γ of trajectories that are most promising to be k-MsTraj. They are discovered by searching through the NTrajI index.

▶ **Proposition 10.** *By construction, k-MsTraj $\subseteq \Gamma \subseteq \mathcal{T}$.*

By exploiting Γ, the baseline method computes the similarity score of each trajectory in Γ and finds the trajectories having the highest similarity with Q within the time interval t.

Therefore, the main task is to construct the *candidate set* Γ using NTrajI. In particular, for a given query trajectory Q, we first restrict query Q within the time query t, obtaining $Q[t]$. Then, for each $(v_i, t_i) \in Q[t]$, we aim at finding the trajectories that are close to v_i within

t_i. To this aim, for each $(v_i, t_i) \in Q[t]$, we search through IT_{v_i}, by NTrajI-search(v_i, t_i) in Algorithm 1, to build $\Gamma_{(v_i,t_i)}$. So we have $\Gamma = \bigcup_{(v_i,t_i)\in Q[t]} \Gamma_{(v_i,t_i)}$. We compute the similarity score with respect to the function in Definition 4, for each trajectory in Γ. In order to maintain the k trajectory ids with the highest similarity score during the search process, we use a heap H, whose top-k entries represent k-MsTraj.

4 Approximated Computation of k-MsTraj

The baseline method described in Section 3 is costly when the number of trajectories in $\mathcal{T}$ is large. To accelerate the searching process, we propose some approximated methods with two-phase preprocessing as discussed in Section 4.1.

- First, we partition G into disjoint groups of nodes, precomputing the distances among the centers of each group. As the similarity function in Definition 4 uses the shortest path distance between nodes of the graph, we aim at approximating distances inside the graph using the distances from the centers of the groups.
- Second, we adapt the NTrajI indexing so that we maintain trajectories among the groups in a structure called VoTrajI.

For the query processing, given a query trajectory, we show how to estimate the similarity scores for the trajectories in $\mathcal{T}$ using the partitioning and the new VoTrajI index in Section 4.2

4.1 Two-phase Preprocessing

The partitioning takes into account node popularity by choosing the nodes having a higher number of trajectories passing through them as the centers of the groups. To do this, it uses Voronoi Diagrams for graphs (VDG), as explained next.

The VDG is a generalization of the classic Voronoi diagram. For graph $G = (V, E)$ and a set of trajectory $\mathcal{T}$, let $\mathcal{C} = \{c_1, c_2, \dots c_h\}$ be a set of h (most popular) nodes in V, called Voronoi sites i.e. center nodes. The VDG over the nodes in $\mathcal{C}$ is defined as a partition of V into h groups $g_1, g_2, \cdots, g_h$, one for each center in $\mathcal{C}$. Node $u \in V$ is in group g_i with center c_i (i.e. $g_i.\mathcal{C} = c_i$) iff $d(u, c_i) \leq d(u, c_j)$ for each $c_j \in \mathcal{C}$ with $i \neq j$ (ties are broken arbitrarily). We can divide G into h Voronoi groups in $O(n \log n)$ time when $h = O(n^\epsilon)$ for a positive constant $\epsilon < 1$ [6].

Once the Voronoi groups have been computed, we precompute and store the pairwise distances among the center nodes of the groups. Our aim is using these distances as an approximation for the distances required by the similarity function in Definition 4. By running one BFS for each center node, we compute the distance between each pair $c_i, c_j \in \mathcal{C}$ in $O(m \cdot n^{1/2})$ time, where we set $h = n^{1/2}$. As a result, we obtain the following lemma.

▶ **Lemma 11.** *Graph partitioning and centers distance precomputation require $O(m \cdot n^{1/2})$ time. The space required by the centers distance table is $O(n)$ space.*

We now discuss how to build the *Voronoi Trajectory Indexing* (VoTrajI) by adapting the NTrajI data structure. We use an interval tree IT_c for each $c \in \mathcal{C}$. Interval tree IT_c stores the time intervals of trajectories in $\mathcal{T}$ spent within the nodes in g, when $g.\mathcal{C} = c$. The VoTrajI maintains the corresponding trajectory ids of the time intervals. By modifying Definition 9, we have:

▶ **Definition 12** (Group Projection Set). *The projection set S_c of a center node $c \in \mathcal{C}$ stores the pairs (t, T) of all trajectories $T \in \mathcal{T}$ that pass through the nodes in g, when $g.\mathcal{C} = c$, namely, $S_c = \{(t, T) \mid (v, t) \in T$ and $v \in g$ and $g.\mathcal{C} = c$ and $T \in \mathcal{T}\}$.*

The Interval Tree IT_c for each node $c \in \mathcal{C}$ maintains all pairs $(t, T) \in S_c$. Each entry of IT_c is the form of $\langle t, id\rangle$, where id is the trajectory id, and t is the time interval that the trajectory id spent at $v \in g$, where $g.\mathcal{C} = c$.

To reduce the storage space used by IT_c, for each $c \in \mathcal{C}$, we consider a sequence of consecutive time intervals with the same trajectory id in S_c as a single time interval with the corresponding trajectory id.

▶ **Lemma 13.** *The two-phase preprocessing takes $O(m \cdot n^{1/2})$ time and $O(n)$ space.*

4.2 Query Processing

Consider how a trajectory $T \in \mathcal{T}$ is represented with respect to the center nodes of the Voronoi diagram. Let $(v_i, t_i) \in T$ and $v_i \in g$, where g is a Voronoi group of G. We represent $(v_i, t_i) \in T$ as (c, t_i) where $g.\mathcal{C} = c$. We obtain a new trajectory T' as a sequence of center nodes and the corresponding time intervals. Note that T' can traverse a sequence of the nodes belonging to the same Voronoi group within consecutive time intervals. To avoid the duplication of nodes for consecutive time intervals, we define *shrunk trajectories*.

Given a trajectory $T' = \langle (c_1, t_1), \ldots, (c_l, t_l)\rangle$, consider the operator $\textsc{shrink}(T')$, which recursively merges any pair $(c_i, t_i), (c_{i+1}, t_{i+1}) \in T'$ as $(c_i, t_i + t_{i+1})$ when $c_i = c_{i+1}$ and t_i, t_{i+1} are two consecutive time intervals (here operation $t_i + t_{i+1}$ gives $[s_i, e_{i+1}]$).

▶ **Definition 14** (Shrunk Trajectory). *Let $T = \langle (v_1, t_1), \ldots, (v_l, t_l)\rangle$ be a trajectory in $\mathcal{T}$. Consider the corresponding sequence $T' = \langle (c_1, t_1), \ldots, (c_l, t_l)\rangle$ with respect to the Voronoi groups. We define the shrunk trajectory of T as $\hat{T} = \textsc{shrink}(T')$.*

Note that it takes $O(l)$ time to obtain $\hat{T}$, and that $|\hat{T}| \leq |T|$. At this point, we consider two variants for estimating k-MSTraj.

SHQ Shrunk Query: Shrinking trajectory Q during query time.

SHQT Shrunk Query and Target: Shrinking each trajectory in $\mathcal{T}$ during the preprocessing and shrinking trajectory Q during query time.

Both variants perform a search on the VoTrajI index using the shrunk query trajectory $\hat{Q}$. The outcome of that search is a set $\tilde{\Gamma}$, which is defined as Γ in Section 3, except that we use VoTrajI in place of NTrajI. This makes a difference, as the property in Proposition 10 does not necessarily hold anymore. Indeed there could be a trajectory $T \in$ k-MsTraj such that $T \notin \tilde{\Gamma}$ (whereas surely $T \in \Gamma$). This approximated version has the advantage of speed, which motivates this study.

4.2.1 Variant SHQ

In this variant we compute the similarity scores for each trajectory $T \in \tilde{\Gamma}$ with respect to the shrunk query $\hat{Q}$. In particular, we make an estimate of $Sim(Q, T, t)$ as $Sim(\hat{Q}, T, t)$, and report the top-k trajectories with the highest estimated similarity score. To measure the precision ratio of this estimation, the similarity function makes an estimate of $d = d(v, u)$ as $\bar{d} = d(c, u)$, when $v \in Q$, $u \in T$, and c is the center node of a group that includes v.

▶ **Lemma 15.** *For any given $v, u \in V$, $u \neq v$, the ratio between $d = dist(v, u)$ and $\bar{d} = dist(c, u)$, where c is the center of Voronoi group containing v, is bounded as $1 \leq d/\bar{d} \leq 3$.*

Proof. Let $dist(c, v) = r$. We have two possibilities. If $r \leq 2\bar{d}$, then by triangle inequality $d \leq r + \bar{d}$ and thus $d \leq 3\bar{d}$. Else if $2\bar{d} < r$, then by triangle inequality $r \leq d + \bar{d}$ and thus $\bar{d} < d$. ◀

Although we reduce the number of nodes in the query trajectory which needs to be processed, the number of distances involved is still large. As mentioned earlier, the cost of distance computation depends on the length of the trajectories within the query time interval t. In order to reduce this cost, we consider our second variant SHQT.

4.2.2 Variant SHQT

In this variant, we estimate $Sim(Q, T, t)$ as $Sim(\hat{Q}, \hat{T}, t)$. Specifically, the similarity function makes an estimate of $d = d(v, u)$ as $\tilde{d} = d(c_i, c_j)$, when $v \in Q$, $u \in T$, and c_i, c_j are the center nodes of the groups that include $v \in Q$ and $u \in T$, respectively.

▶ **Lemma 16.** *For any two distinct nodes v, u belonging to Voronoi groups with center nodes c_i, c_j, respectively, the ratio between $\tilde{d} = dist(c_i, c_j)$ and $\bar{d} = dist(c_i, u)$ is bounded as $\tilde{d}/\bar{d} \leq 2$.*

Proof. As illustrated in Figure 2, let $dist(v, c_i) = r$ and $dist(u, c_j) = r'$. We consider the groups g_i, g_j containing the two centers c_i, c_j, respectively. By triangle inequality we have $\tilde{d} \leq r' + \bar{d}$. Since $u \in g_j$ and $u \notin g_i$ then $r' \leq \bar{d}$. Thus, $\tilde{d} \leq 2\bar{d}$. ◀

Using Lemma 15 and 16, we are able to conclude that $\tilde{d} \leq 2d$ when $r > 2\bar{d}$. If $r > 2\bar{d}$ by triangle inequality we have $\bar{d} < d$. Since $\tilde{d} \leq 2\bar{d}$, we can obtain that $\tilde{d} \leq 2d$.

The similarity function in Definition 4 assigns a larger contribution to those nodes of the trajectories that are closer to each other, rather than the farther ones. Thus, by Lemma 15 and 16, we expect that the estimated similarity score in both variants is larger than the exact similarity score. We will evaluate this idea in our experiments.

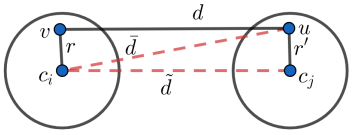


Figure 2 $c_i \in g_i$ and $c_j \in g_j$ such that $c_i, c_j \in \mathcal{C}$.

Table 2 Summary of Datasets. Recall that D_G is the diameter of G.

Dataset Name	trajectories	nodes	edges	D_G
Facebook	1000	4039	88234	8
Milan	16166	3000	130071	5
Rome	7755	473	10524	6

5 Experimental Evaluation

This section is devoted to comparing the performances of SHQ and SHQT with respect to the baseline method, hereafter called BASE. The evaluation aims at providing a response to the following questions.

Q1: How fast is getting the answer for a query, i.e. how much is the query time?

Q2: How fast is the preprocessing phase?

Q3: How good is the quality of the solution found if compared with the exact solution?

Table 3 The average time for answering a query for each method (in sec).

Datasets	BASE	SHQ	SHQT
Facebook	1.09	0.74	0.43
Milan	380.03	376.15	85.48
Rome	26.19	19.42	15.83

Table 4 The average number of trajectories in the candidate set in each method. For the Facebook dataset, refer to Figure 4.

Datasets	BASE	SHQ	SHQT
Milan	9786.39	9968.98	9968.98
Rome	7504.37	6569.84	6569.84

We will respond to these questions by evaluating the performance of each method for different value of k. In particular, we set k as 2^i for $i = 0, \ldots, 6$. Each experiment requires a graph, a trajectory set, and a query trajectory. For each experiment, we choose 100 trajectories as query trajectories, randomly.

Our computing platform is a machine with Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40GHz, 24 virtual cores, 128 Gb RAM, running Ubuntu Linux version 4.4.0-22-generic. The source code has been written in `Python3`.

5.1 Datasets

We conduct our experiments on real-world graphs, using synthetic and real trajectories, whose properties are shown in Table 2.

Facebook Synthetic trajectories from Facebook social network. [2]

Milan Dataset based on GPS tracks of private cars in Milan. First, we build the graph by making use of the GPS trajectories. Then, we cluster the close nodes with k-Means. There is an edge between two clusters i and j if there exists at least one trajectory going through i, j, consecutively. [3]

Rome Dataset of Flickr geo-tagged photos provided by [10], containing tourist trajectories covering Rome. We build the graph of the Points of Interest (in short, PoIs).

5.2 Query Time

In the following, we compare the query time of the three methods. Table 3 reports our results, showing the average query time over 100 queries in each dataset. As it can be seen, both SHQ and SHQT variants outperform BASE. The most evident benefit can be seen for the biggest dataset that we considered, i.e. Milan dataset. In this case, SHQT spends less than 23% of the time needed by BASE and SHQ.

We report in Table 4 the number of candidate trajectories for all the methods (i.e. $|\Gamma|$ and $|\tilde{\Gamma}|$). The table shows that SHQ and SHQT select more candidates than BASE. This is not an issue as, even if we have to process these extra trajectories, they compute much fewer query distances and consecutively spend less time than BASE, as shown in Table 3.

For the sake of completeness, we also analyzed the behavior of our method when the length of the queried trajectories varies for the case of the Facebook dataset. As we can see in Figure 3, SHQ and SHQT outperform BASE. Actually SHQT significantly outperforms the other two, and the improvement becomes even more evident when the length of the query trajectory increases. As the length of the query goes up to 80, the time needed by BASE

[2] `https://snap.stanford.edu/data/ego-Facebook.html`
[3] `https://sobigdata.d4science.org/catalogue-sobigdata?path=/dataset/gps_track_milan_italy`

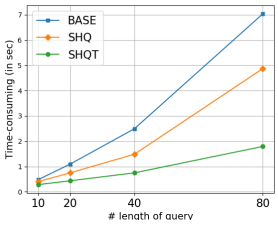


Figure 3 Average time vs length of queried trajectories in Facebook dataset.

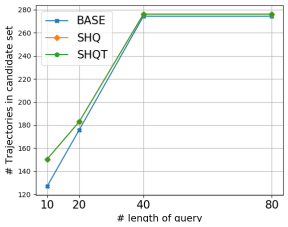


Figure 4 The average number of trajectories in candidate set in each method vs length of query in Facebook dataset. For both SHQ and SHQT the candidate set $\tilde{\Gamma}$ is the same.

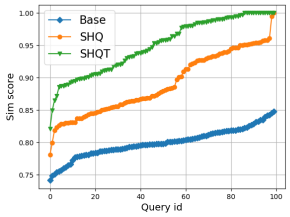


Figure 5 The comparison between similarity scores of each method.

and SHQ increases faster than the SHQT. This observation confirms that shrinking both target and query trajectories reduces the number of distance computations and thus the time reduced greatly.

Figure 4 shows the average number of trajectories in the candidate set for each method. Note that the number of trajectories in the candidate set for both methods SHQ and SHQT is the same since we use the same approach for specifying $\tilde{\Gamma}$. As it can be seen, by increasing the length of the query up to 40, the number of trajectories in the candidate set for each method increases quickly to more than 240, and then becomes the same for all of them. This confirms the role of the precomputed distances among Voronoi centers to accelerate query processing. The time cost of this precomputation is negligible. Moreover, by Lemmas 15 and 16, we would expect that the similarity score would be larger when we shrink trajectories: looking at Figure 5, we observe that the similarity scores by shrinking trajectories behave as we expected.

5.3 Preprocessing Time

As we mentioned before, between SHQT and SHQ, only the former uses the precomputed distances. The cost of the precomputation is shown in Table 5.We also report the time needed to index and shrink trajectories. In particular, columns NTrajI and VoTrajI report the time needed for building respectively the indexing structures NTrajI and VoTrajI. As expected, the time needed for building NTrajI over the trajectories in the Facebook dataset is larger than the one required by other datasets. This is due to the presence of longer trajectories with respect to other datasets. The column Distance Precomputing shows that the time needed to precompute the distances is negligible. Finally, we can observe that the time needed to perform the Voronoi partitioning and shrinking trajectories in the last column is also negligible in comparison with the time needed for building NTrajI, which clearly dominates the cost.

5.4 Quality and Evaluation Metrics

We evaluate the quality of the solution produced by the SHQ and SHQT methods with respect to BASE. The effectiveness of the methods is assessed by means of the metrics that we describe next, where the values close to 1 are more desirable. Let γ_1 and γ_2 be two output sets containing top-k trajectories, e.g. the exact and approximated solutions.

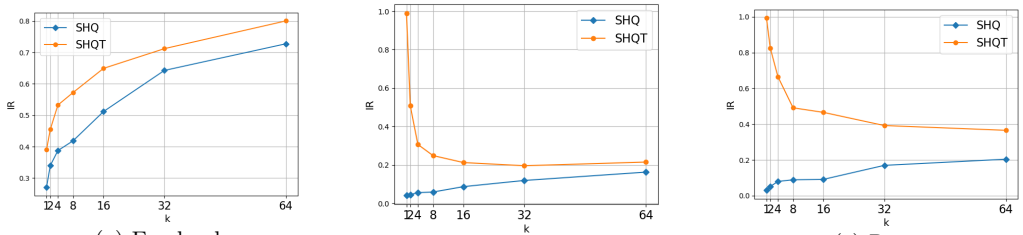
Table 5 Preprocessing time (in sec).

Dataset	NTrajI	VoTrajI	Distance precomputing	Shrinking trajectories and building Voronoi diagram
Facebook	185.19	0.51	0.48	0.62
Milan	1716.44	13.50	0.27	10.21
Rome	69.25	0.24	0.005	0.56

1. We define the *similarity score ratio* as the ratio of the average similarity scores of trajectories in γ_1 and γ_2. Namely, $\text{SSR}(\gamma_1, \gamma_2) = \frac{\sum_{T \in \gamma_1} Sim(Q,T,t)}{\sum_{S \in \gamma_2} Sim(Q,S,t)}$.
2. We define the *intersection ratio* as $\text{IR}(\gamma_1, \gamma_2) = \frac{|\gamma_1 \cap \gamma_2|}{k}$.

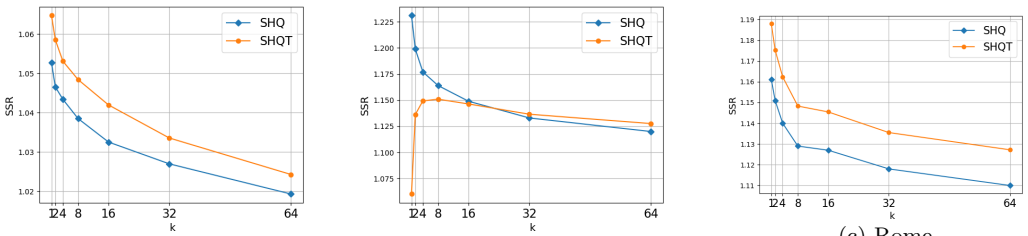
It is worth remarking that the running time of the methods does not change for the different values of k.

Our results are shown in Figures 6 and 7, where the IR and SSR ratios are reported as a function of k. In particular, Figure 6 represents the IR ratio for increasing k values in the three datasets. The IR ratio goes up to more than 0.80 quickly, by increasing the value of k on the Facebook network. On the other hand, this value in Milan and Rome networks becomes close to 0.3. However, we observe that the lower values of IR correspond to SSR values that are close to 1. Indeed, Figure 7 shows SSR which is almost always very close to 1 and that gets more close to 1, while increasing k.



(a) Facebook (b) Milan (c) Rome

Figure 6 The quality of the results returned by the competitors in terms of IR ratio vs k.



(a) Facebook (b) Milan (c) Rome

Figure 7 The quality of the results returned by the competitors in terms of SSR ratio vs k.

References

1 Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer, 1993.

2 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag TELOS, 2008.

3 Lei Chen and Raymond Ng. On the marriage of Lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803. VLDB Endowment, 2004.

4 Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM, 2005.

5 Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. Searching trajectories by locations: an efficiency study. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 255–266. ACM, 2010.

6 Martin Erwig. The graph Voronoi diagram with applications. *Networks: An International Journal*, 36(3):156–163, 2000.

7 Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of ACM SIGMOD*, pages 419–429, Minneapolis, MN, 1994.

8 Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. Searching for similar trajectories on road networks using spatio-temporal similarity. In *East European Conference on Advances in Databases and Information Systems*, pages 282–295. Springer, 2006.

9 Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

10 Cristina Ioana Muntean, Franco Maria Nardini, Fabrizio Silvestri, and Ranieri Baraglia. On learning prediction models for tourists paths. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(1):1–34, 2015.

11 Shuo Shang, Ruogu Ding, Kai Zheng, Christian S Jensen, Panos Kalnis, and Xiaofang Zhou. Personalized trajectory matching in spatial networks. *The VLDB Journal*, 23(3):449–468, 2014.

12 Eleftherios Tiakas, Apostolos N Papadopoulos, Alexandros Nanopoulos, Yannis Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. Trajectory similarity search in spatial networks. In *null*, pages 185–192. IEEE, 2006.

13 Eleftherios Tiakas and Dimitrios Rafailidis. Scalable trajectory similarity search based on locations in spatial networks. In *Model and Data Engineering*, pages 213–224. Springer, 2015.

14 Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684. IEEE, 2002.

15 Jung-Im Won, Sang-Wook Kim, Ji-Haeng Baek, and Junghoon Lee. Trajectory clustering in road network environment. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 299–305. IEEE, 2009.

16 Ying Xia, Guo-Yin Wang, Xu Zhang, Gyoung-Bae Kim, and Hae-Young Bae. Spatio-temporal similarity measure for network constrained trajectory data. *International Journal of Computational Intelligence Systems*, 4(5):1070–1079, 2011.